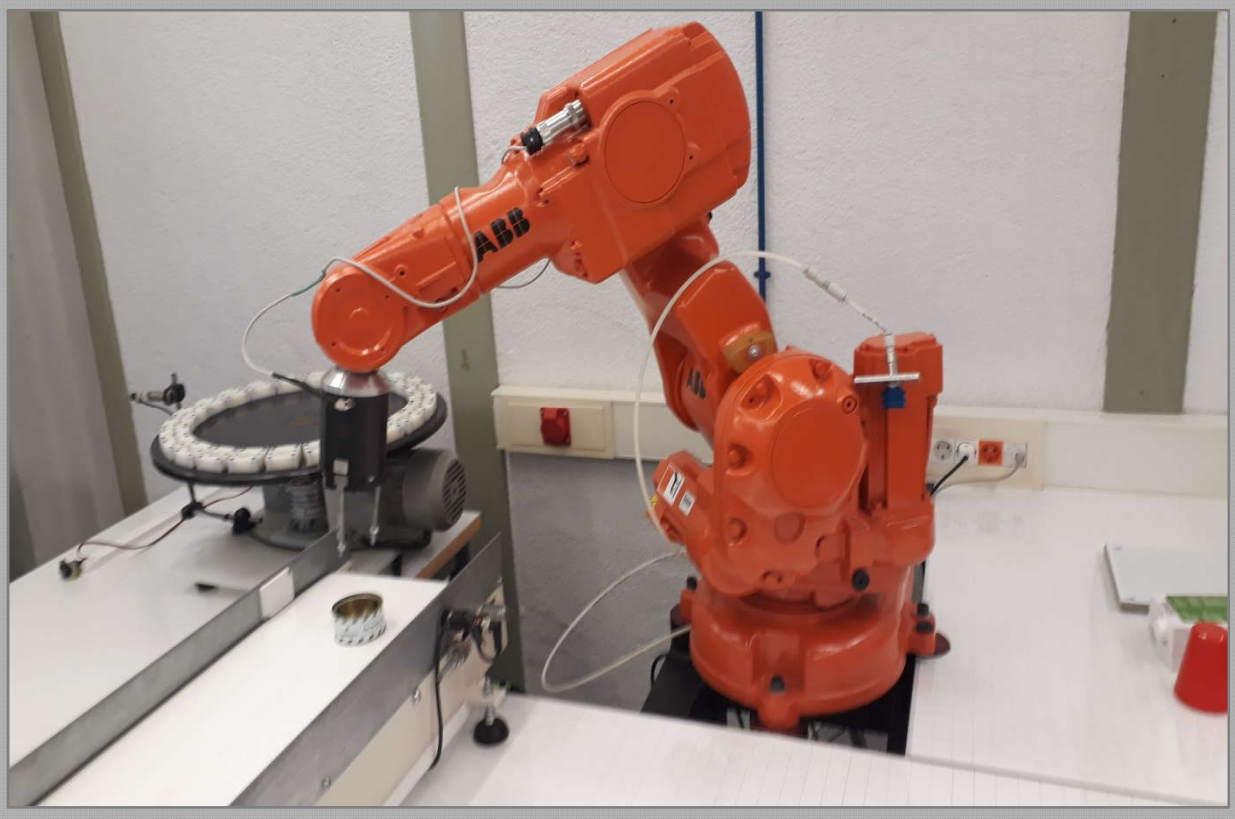
TRABAJO FIN DE GRADO

PROGRAMACIÓN DE UNA CÉLULA ROBOTIZADA PARA LA MANIPULACIÓN DE LATAS DE CONSERVA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TUTOR: Carlos Ricolfe Viala

ALUMNO: Javier Gracia Andrés

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ETSID

UNIVERSIDAD POLITÉCNICA DE VALENCIA

PROGRAMACIÓN DE UNA CÉLULA ROBOTIZADA PARA LA MANIPULACIÓN DE LATAS DE CONSERVA

DOCUMENTOS:

1. **MEMORIA**
2. **PRESUPUESTO**

Autor: Javier Gracia Andrés

Tutor: Carlos Ricolfe Viala

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

ETSID

UNIVERSIDAD POLITÉCNICA DE VALENCIA

PROGRAMACIÓN DE UNA CÉLULA ROBOTIZADA PARA LA MANIPULACIÓN DE LATAS DE CONSERVA

DOCUMENTO:

MEMORIA

Autor: Javier Gracia Andrés

Tutor: Carlos Ricolfe Viala

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

ÍNDICE

ÍNDICE DE ILUSTRACIONES

MEMORIA

1. OBJETIVO

El objetivo del presente proyecto se trata de la programación de una célula robótica que permita la discriminación por posición de latas de conserva.

Se debe conseguir el reconocimiento de la lata mediante el uso de la cámara de visión artificial JAI CV-M77, de forma que se pueda distinguir si la lata tiene la obertura visible u oculta.

Una vez se haya realizado el reconocimiento visual de la lata el robot IRB 140 se encargará de coger la lata. Si la obertura es visible llevará la lata para el inicio del proceso industrial, si la obertura está oculta la llevará a un sistema que conseguirá dejar la obertura visible.

2. INTRODUCCIÓN

En las últimas décadas se están incluyendo robots en los procesos ya que aceleran el proceso industrial y disminuyen costes a la hora de realizar los productos.

Es posible programar los robots con distintos lenguajes de programación de forma que realicen las tareas sin interactuar con el ser humano. Para realizar las distintas tareas a tiempo real atendiendo a las variables del proceso necesitan apoyarse en dispositivos que reconozcan dichas variables, como pueden ser sensores, cámaras de visión artificial, etc.

En los últimos años las cámaras de visión artificial están siendo muy utilizadas ya que reconocen y ofrecen mucha información de los objetos, son capaces de distinguir fallos en el objeto, determinar su posición, orientación e innumerables datos que facilitan y ofrecen muchas alternativas a la hora de programar los robots.

En el TFG desarrollado en este documento se ha conseguido distinguir y alcanzar las latas de conserva automáticamente independientemente de la posición de la lata sobre la cinta transportadora y además se distinguirá si la lata tiene la obertura visible u oculta.

3. SISTEMAS DE LA SOLUCIÓN ADOPTADA

La solución adoptada consta de distintos sistemas, los cuales desarrollaremos a continuación.

3.1 ROBOT IRB 140

Cuenta con 6 ejes, soporta una carga de 6 Kg y tiene un alcance de 810 cm. Utilizaremos la herramienta que permite el agarre de la pieza con los dedos colocados en el extremo del eje 6.

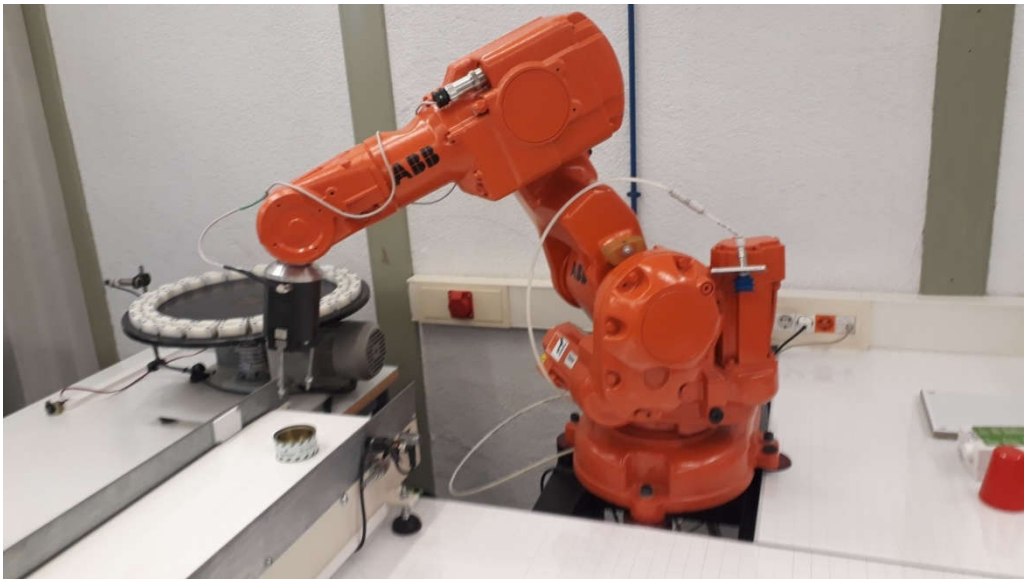


Ilustración 1. Robot IRB 140

El robot cuenta con el armario donde se encuentran el controlador IRC5, puerto USB y Ethernet, entradas y salidas (tanto analógicas como digitales).

Ilustración 2. Armario Robot

El robot cuenta con la herramienta Flex Pendant para el control de operaciones del robot, ejecución de los programas, etc.

Ilustración 3. Flex Pendant

3.2 CINTA TRANSPORTADORA

Este sistema mueve la lata desde un extremo donde caerán las latas y las transportará hasta el otro extremo en el que se encuentra el sensor que detendrá el avance para el análisis de la imagen y la recogida de la propia lata.

Ilustración 4. Cinta transportadora

3.3 CÁMARA VISIÓN ARTIFICIAL JAI CV-M77

Se encarga del reconocimiento de la pieza sobre la superficie plana de la cinta transportadora.

Ilustración 5. Cámara JAI CV-M77

3.4 ORDENADOR INDUSTRIAL INFAMOIN

Contiene el programa Sherlock 7, se puede acceder a él desde la *conexión a escritorio remoto* de cualquier ordenador.

Ilustración 6. Ordenador industrial Infamoin

3.5 LATA DE CONSERVA

Se trata del objeto a manipular.

Ilustración 7. Lata de conserva

Las dimensiones de la lata se pueden observar en la siguiente ilustración.

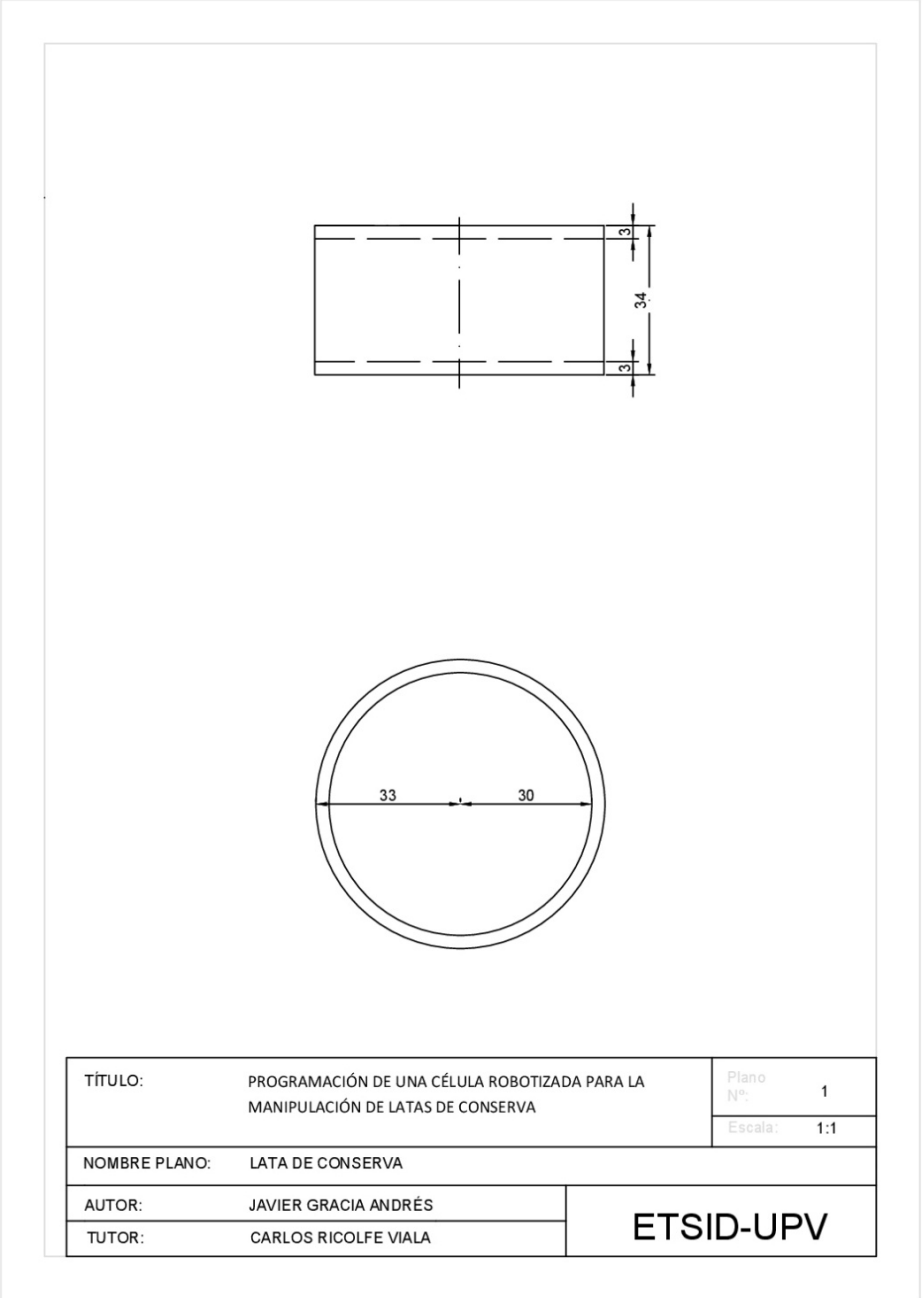


Ilustración 8. Plano lata

4. DESARROLLO SOLUCIÓN ADOPTADA

En el siguiente apartado se va a desarrollar la solución que se ha trabajado para alcanzar el objetivo del proyecto.

4.1 OBTENCIÓN DE LOS PUNTOS DEL PATRÓN

La calibración se realiza para conseguir que el robot y la cámara compartan el mismo sistema de coordenadas. El programa Sherlock 7 que utiliza la cámara permite calibrar la cámara con el uso de un patrón de calibración.

El patrón utilizado para la calibración es el siguiente:

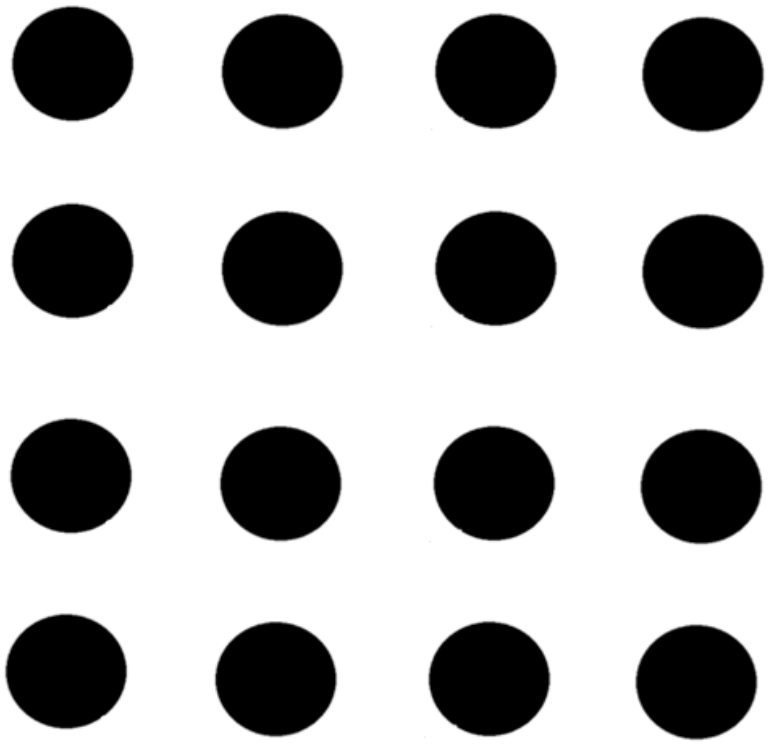


Ilustración 9. Patrón de calibración

Una vez tengamos el patrón de calibración que deseamos se debe colocar sobre el plano de trabajo, en nuestro caso la cinta transportadora, y situar el robot encima de cada punto para anotar su posición central, como se indica en las siguientes ilustraciones.

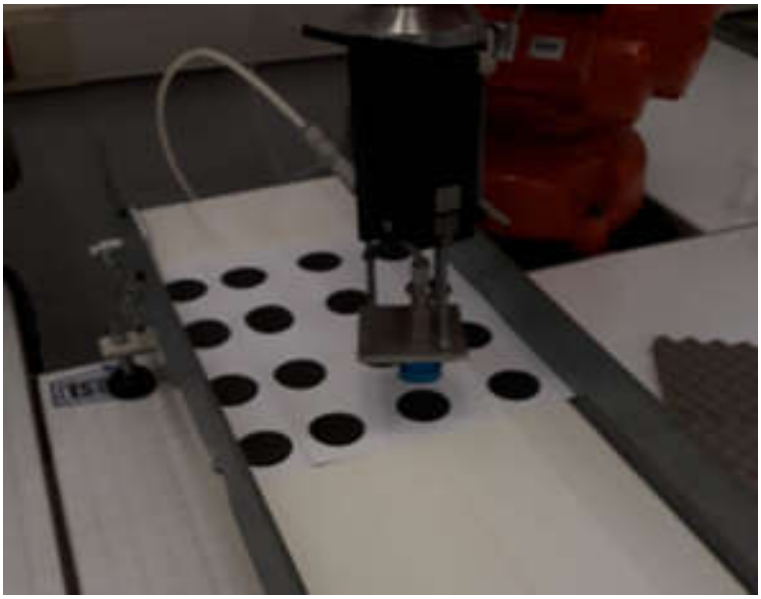


Ilustración 10. Medida Puntos

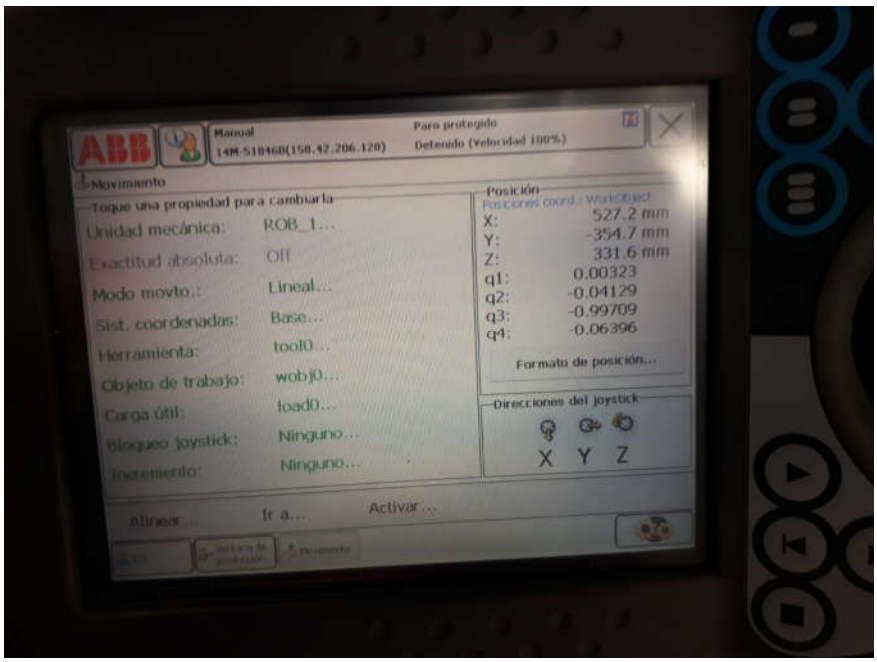


Ilustración 11. Coordenada puntos calibración

4.2 CALIBRACIÓN SHERLOCK 7

A continuación, con el programa Sherlock 7 se crea la calibración desde la opción *calibration* pulsando el botón *Add*.

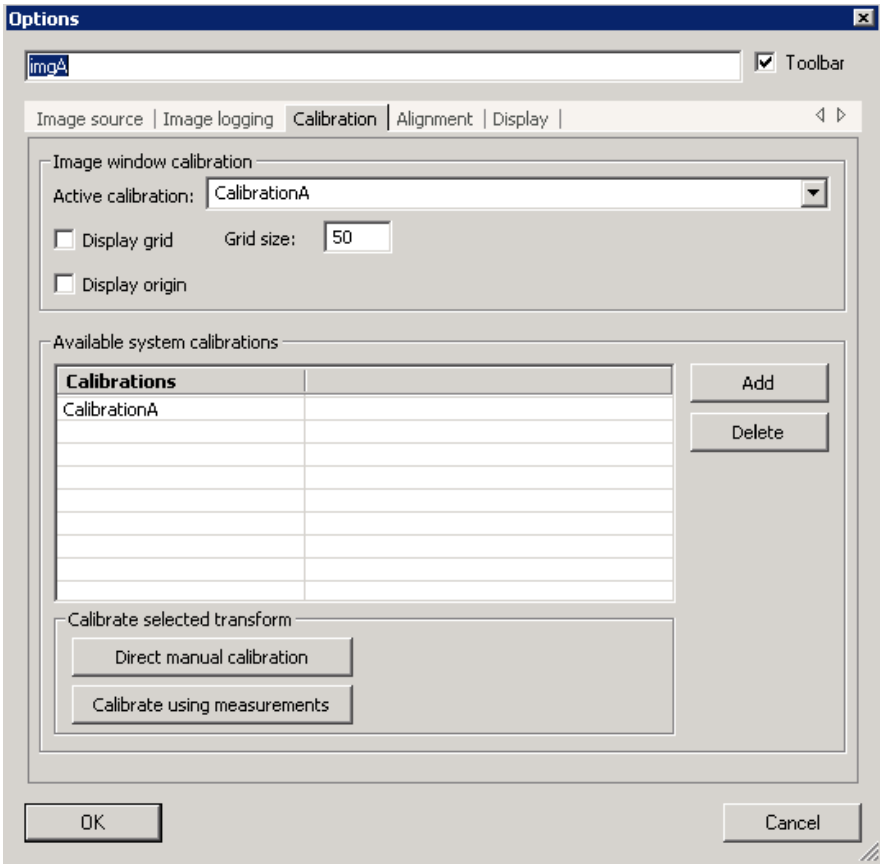


Ilustración 12. Calibración Sherlock 7

Seleccionando en el menú el botón *calibrate using measurements* podemos visualizar el patrón de calibración desde la cámara, para ello debemos pulsar sobre el centro de los puntos con el cursor y se introducirá la coordenada (X,Y) de cada punto obtenida antes con el robot.

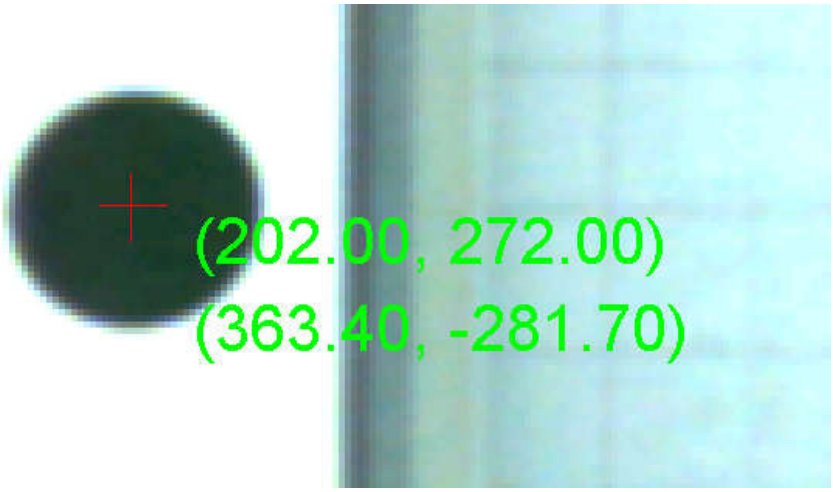


Ilustración 13. Punto calibración

4.3 PROGRAMACIÓN DEL PROGRAMA SHERLOCK 7

Con la programación de Sherlock 7 se ha conseguido el reconocimiento de las piezas, la conexión con el robot, inicializar las variables y un diálogo que permite ver si la cinta esta parada.

4.3.1 CONEXIÓN CÁMARA-ROBOT

La conexión cámara-robot se realiza mediante el protocolo de comunicación TPC/IP. Para programar este tipo de conexión se debe seleccionar el desplegable *Options* y seleccionar *IO configuration*. Una vez dentro se pulsa sobre *TPC/IP* y se selecciona la cámara como servidor y el puerto que vamos a utilizar, en este caso el 1027.

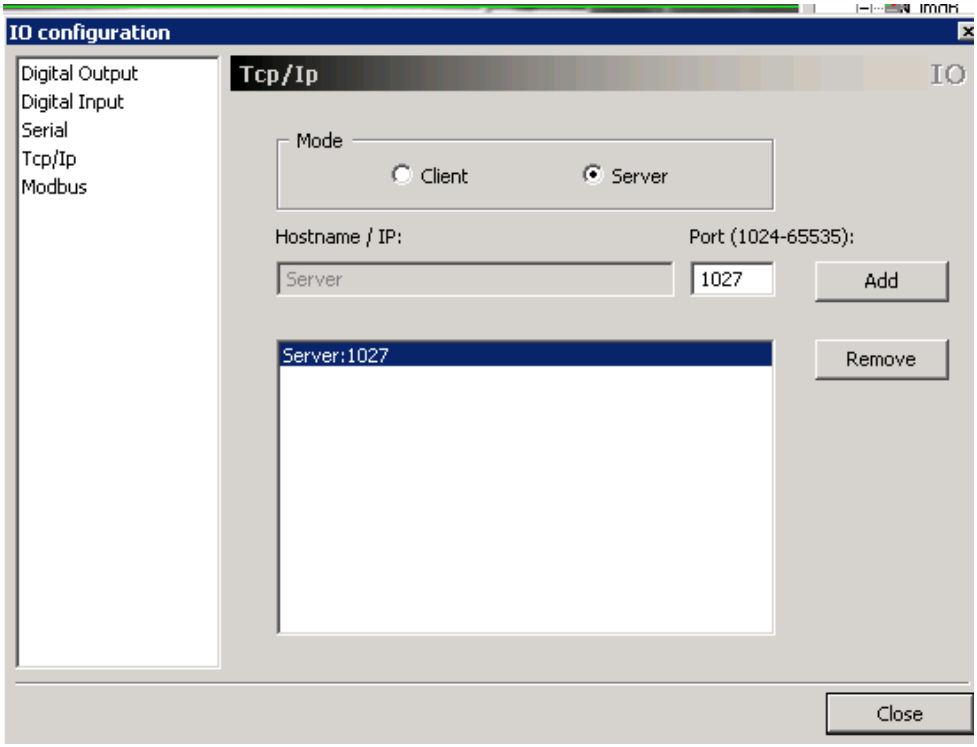


Ilustración 14. Configuración Tpc/Ip

Una vez terminados los pasos anteriores se puede realizar el apartado de conexión que nos permitirá visualizar un dialogo entre la cámara y el robot.

Para llevar a cabo la conexión se utilizan dos bloques: *send line* y *recv line*.

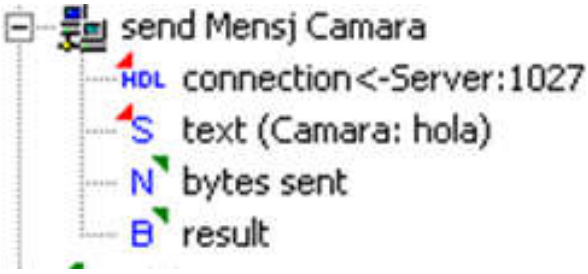


Ilustración 15. Bloque *send line*

Ilustración 16. Bloque *recv line*

A ambos bloques se les asigna el puerto 1027 para la conexión.

Al bloque *revc line* se le asigna el termino ASCII 33, correspondiente al carácter "!", que cerrará el dialogo cuando reciba el carácter mencionado. El tiempo de espera para recibir el mensaje es de 1 minuto, que corresponde a 60000.00 ms.

En la siguiente Ilustración se muestra la subrutina de conexión:

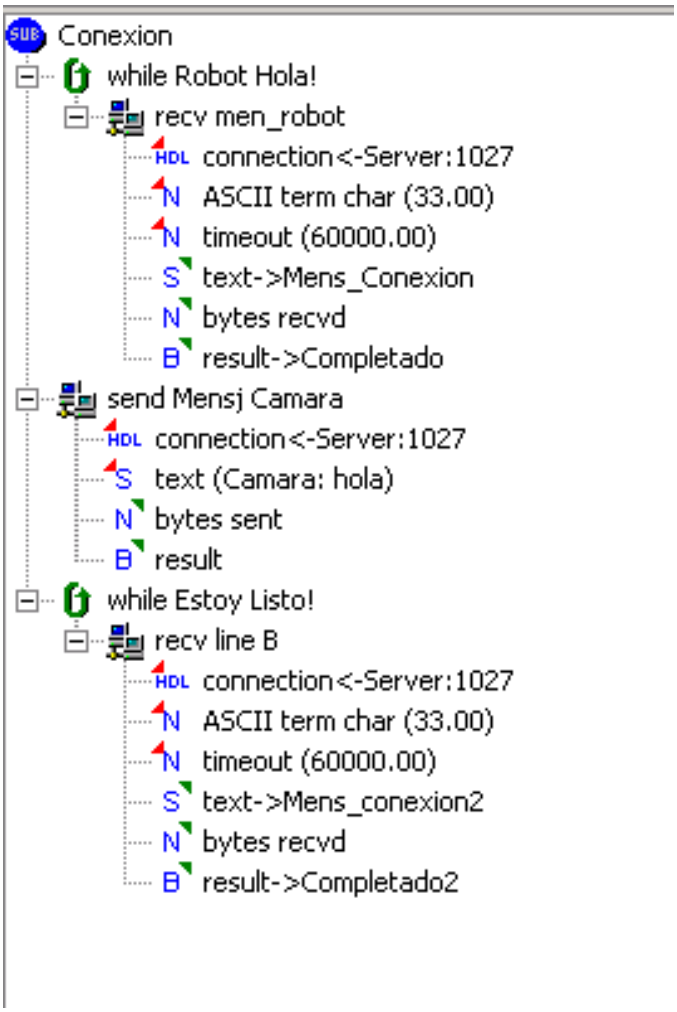


Ilustración 17. Subrutina conexión

4.3.2 INICIALIZACIÓN DE LAS VARIABLES

Esta subrutina del programa se utiliza para inicializar las variables antes de ejecutar el programa principal, de modo que estas no contengan valores de ejecuciones anteriores.

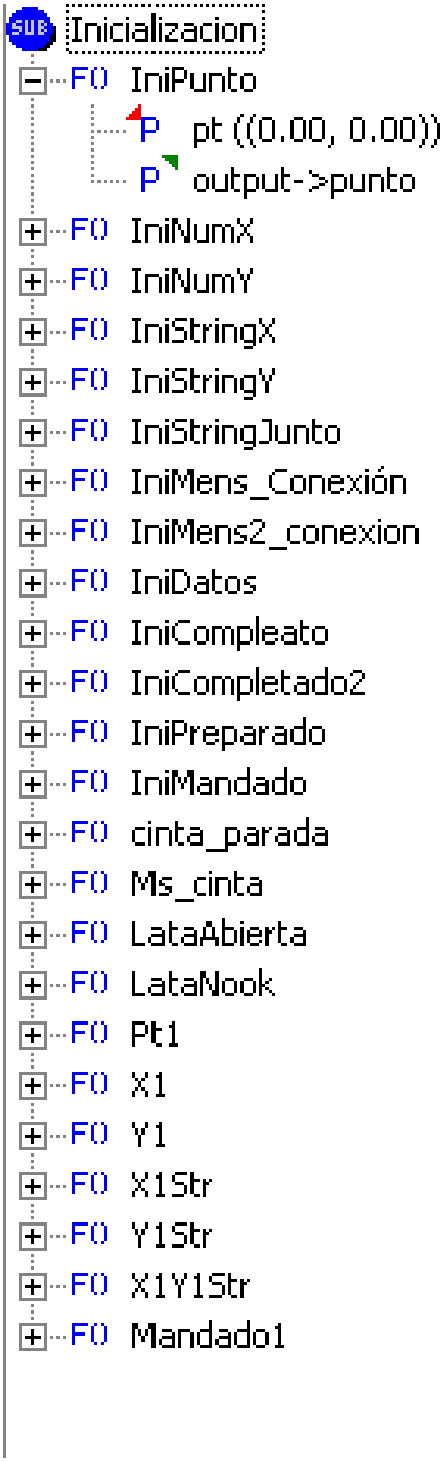


Ilustración 18. Subrutina Inicialización

4.3.3 CINTA PARADA

Esta subrutina se trata de un diálogo entre el robot y Sherlock 7 que nos permite observar si la cinta se ha parado cuando la lata pasa por el sensor de posición.

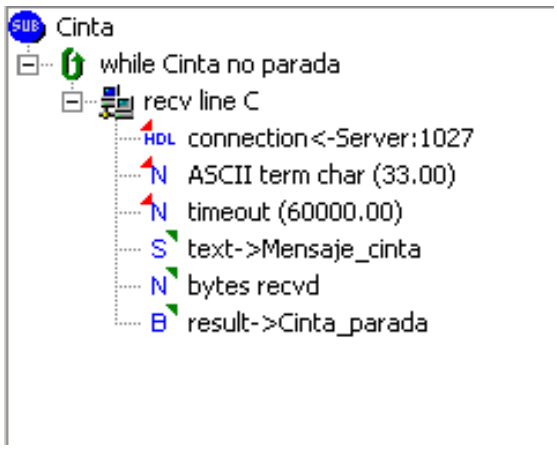


Ilustración 19. Subrutina Cinta

4.3.4 PROGRAMA PRINCIPAL

Mediante el programa principal se llama a los bloques mencionados anteriormente, se realiza el tratamiento de imagen y se encuentran las instrucciones que permiten enviar al robot la posición de la pieza en formato *String*.

4.3.4.1 TRATAMIENTO DE LA IMAGEN

Para reconocer la lata se ha de abrir dos imágenes respecto de la imagen A, que es la que ve la cámara en tiempo real, y aplicar el filtro *Mono 8*. Las dos imágenes serán la B y C y se utilizará cada una para reconocer si la lata está con la obertura visible u oculta.

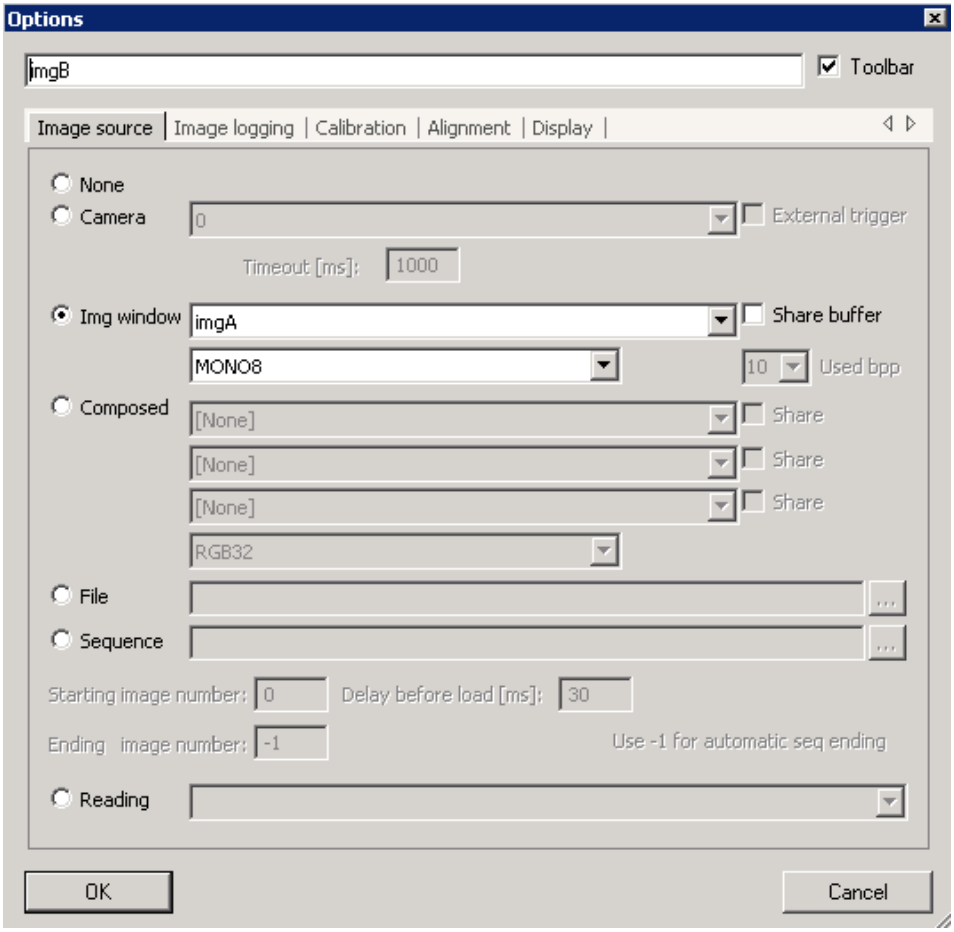


Ilustración 20. Apertura imágenes B y C

A continuación, se sitúa la lata junto al sensor de la cinta, puesto que va a ser el lugar donde el robot recoja la lata, y se crea un *ROI* que contenga la lata. Para continuar se añade el pre procesado *Threshold*, que nos permite binarizar la imagen de forma que quede el objeto de color negro y el fondo blanco. Finalmente se aplica el algoritmo *Search Geometric*, el cual reconoce el contorno del objeto y sus características.

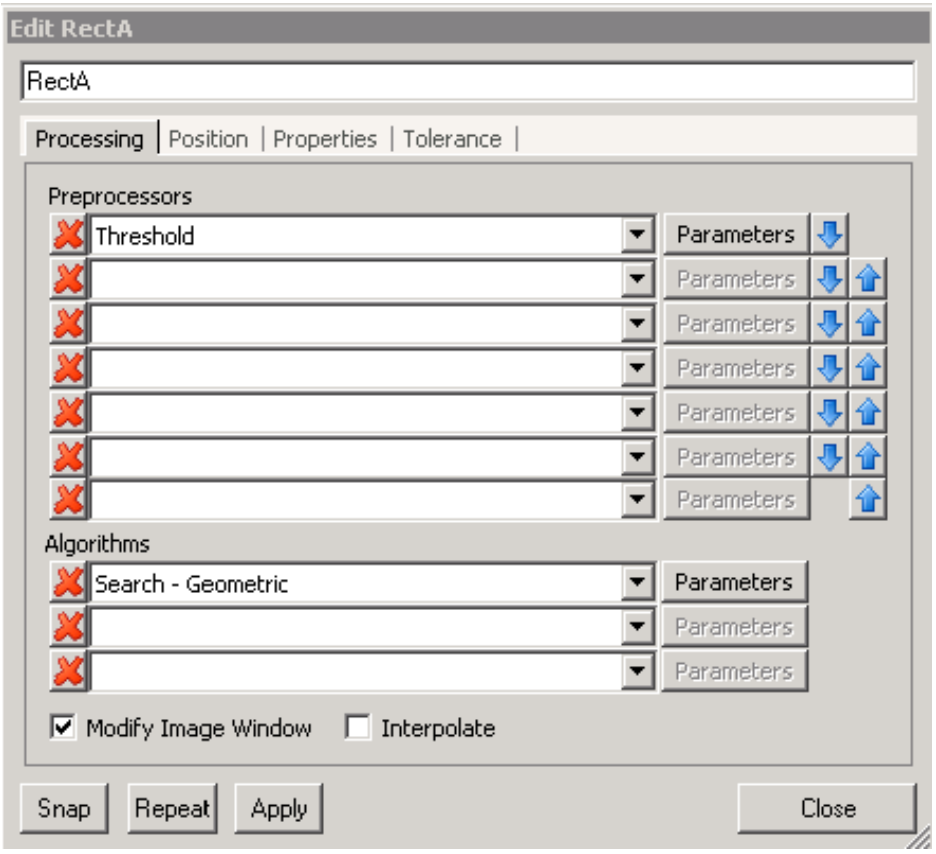


Ilustración 21. Programación ROI

En el pre procesado *Threshold* se debe ajustar los parámetros una vez que la iluminación del laboratorio sea total y de forma que se obtenga la mayor cantidad de información que permita diferenciar la posición de la lata.

Se ha modificado el parámetro *Threshold* a 190 para los dos *ROI* de ambas imágenes, este parámetro varía desde 256, que deja el *ROI* completamente negro, hasta 0, donde el *ROI* queda completamente blanco.

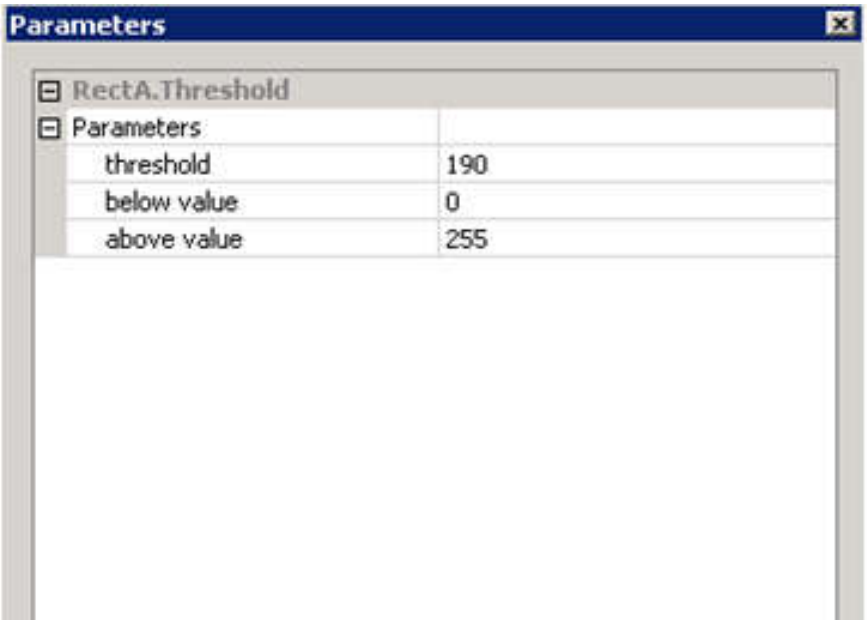


Ilustración 22. Parámetros *Threshold*

El algoritmo *Search Geometric* marca puntos de color verde entre los bits contiguos y distintos entre sí, es decir, marca un punto cuando existe un bit a 0 contiguo a un bit que este a 1. A su vez nos permite eliminar puntos del contorno mostrándolos en color rojo, de forma que podemos elegir qué puntos de la pieza nos convienen más para nuestra solución.

Una vez se han seleccionado los puntos del contorno que nos conviene, se ha de ajustar el parámetro *min score* que nos permite reconocer la posición de la lata si el parecido con la imagen tratada es igual o superior al porcentaje elegido.

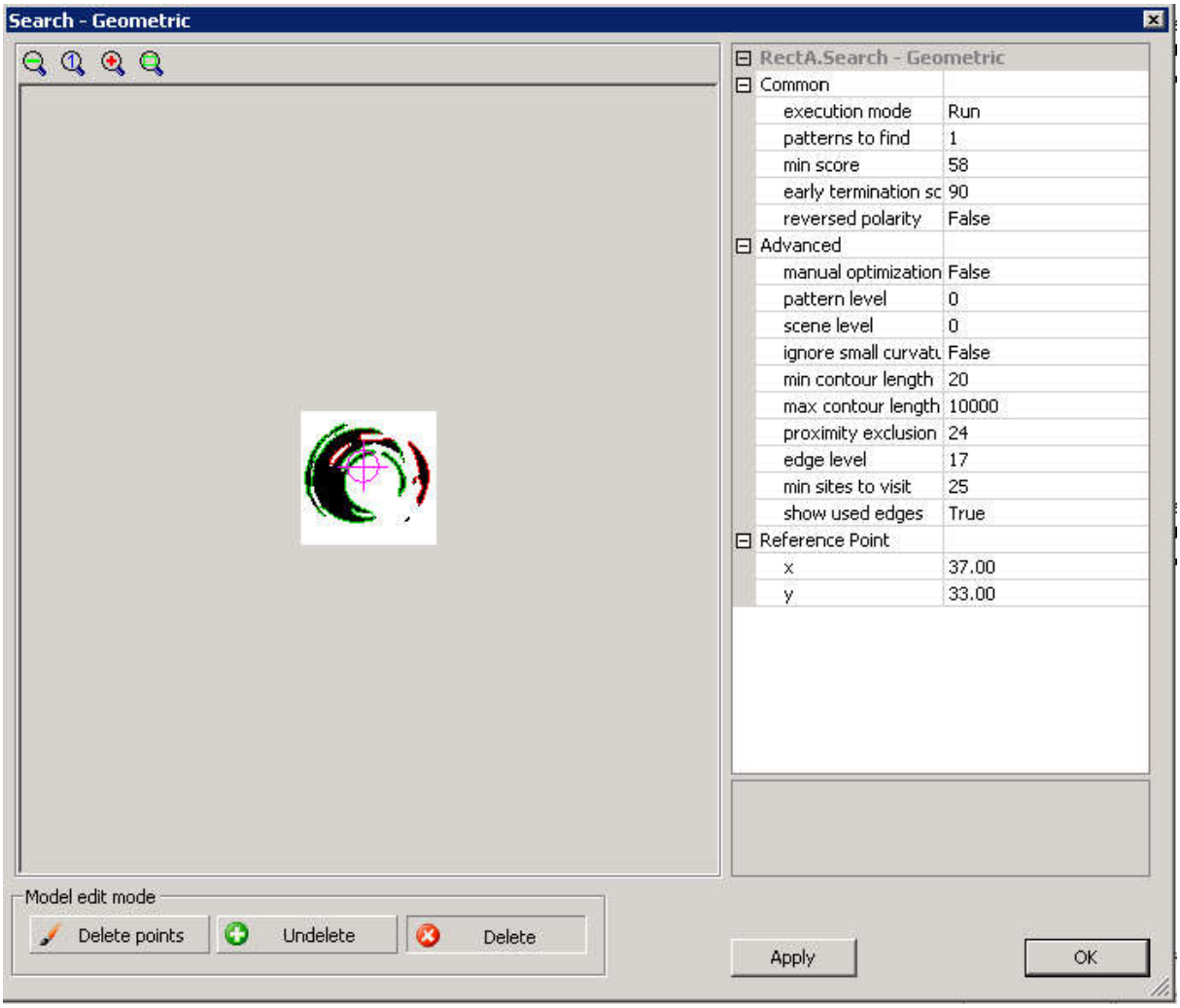


Ilustración 23. Parámetros *Search Geometric*

4.3.4.2 INSTRUCCIONES DEL PROGRAMA PRINCIPAL

Para enviar las coordenadas al robot se debe mandar en formato *String*, para ello se utilizan instrucciones que permiten convertir la coordenada en formato Point a formato *String*.

En primer lugar, se separa la coordenada en formato Point en dos variables numéricas X e Y mediante la instrucción *PtToXY*.

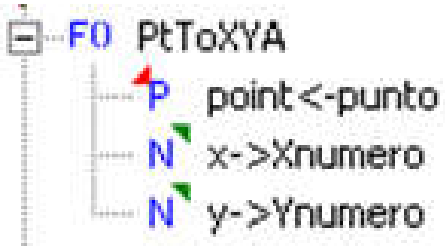


Ilustración 24. Instrucción *PtToXY*

En segundo lugar, se convierten las variables numéricas obtenidas X e Y en variables *String* utilizando la instrucción *PrintfNum*.

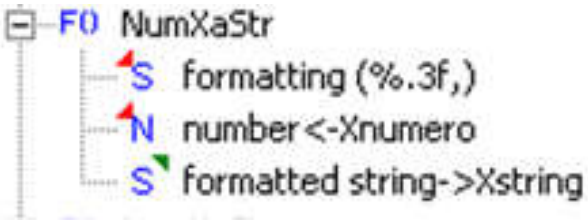


Ilustración 25. Instrucción *PrintfNum X*

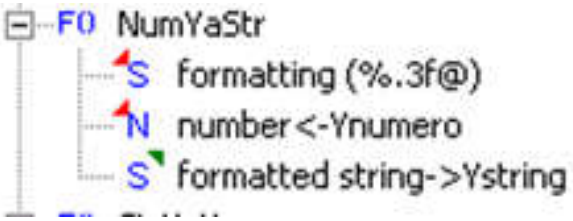


Ilustración 26. Instrucción *PrintfNum Y*

Se utiliza el formato *%f* asignándole 3 decimales como se muestra en la ilustraciones 24 y 25.

Tras la conversión de la variable numérica X se le añade una coma y al juntarla con la variable Y queden ambas con el formato (X,Y).

El carácter @ se añade tras la conversión de la variable numérica Y cuando la lata tenga la obertura visible, y cuando la obertura este oculta se añadirá el carácter #.

En último lugar, se juntan las variables ya en formato *String* usando la instrucción *AddStr*. La variable resultante que devuelva la función será la que se utilice para mandar la coordenada al robot.

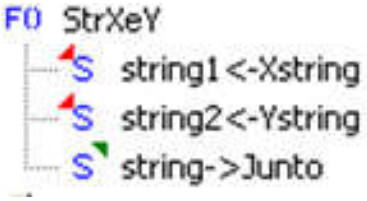
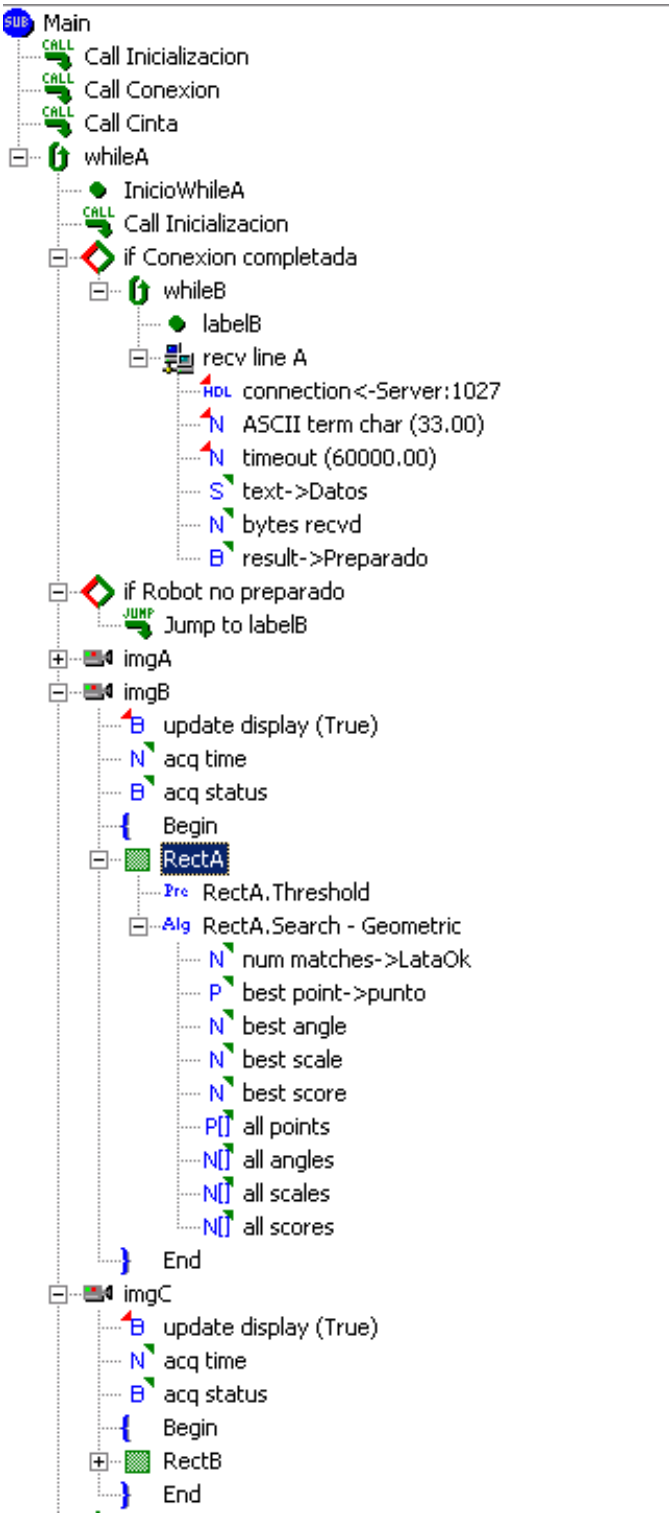


Ilustración 27. Instrucción *AddStr*

4.3.4.3 ESTRUCTURA PROGRAMA PRINCIPAL



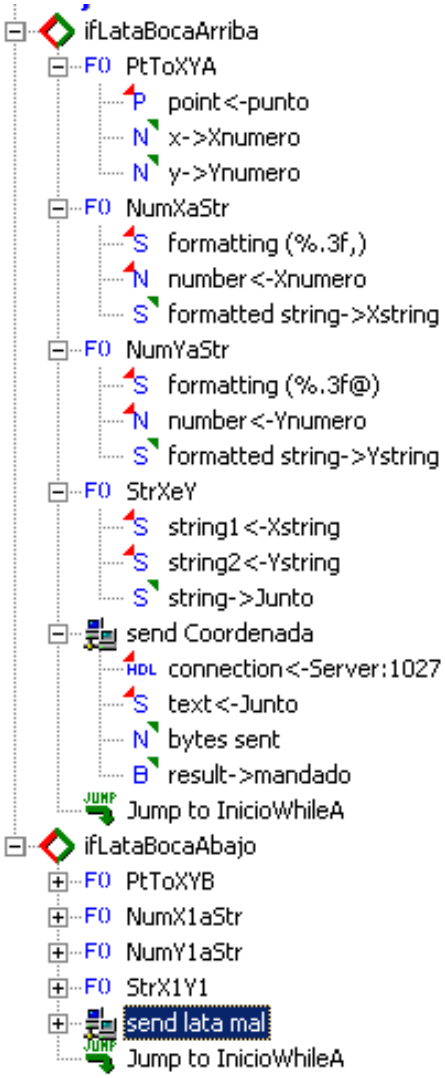
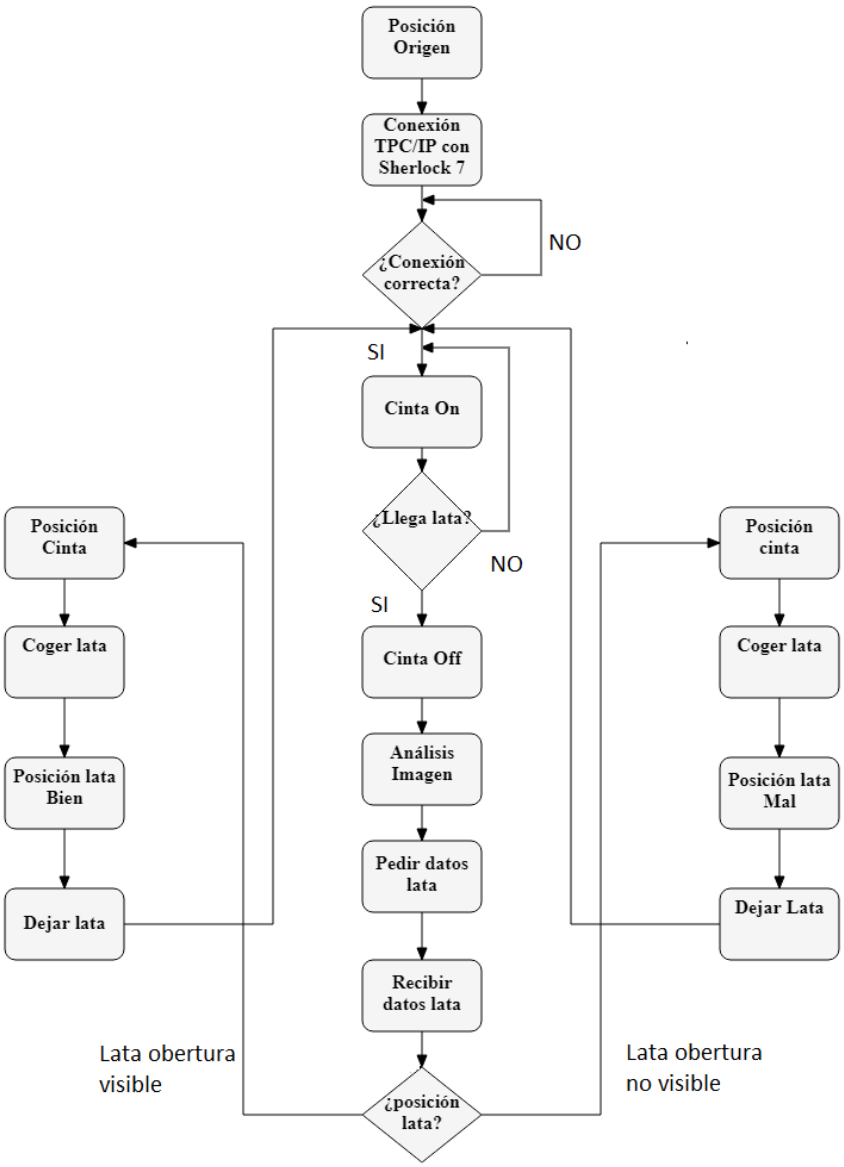


Ilustración 28. Programa principal Sherlock 7

4.4 PROGRAMACIÓN DEL IRB 140

Los robots ABB utilizan el programa Robot Studio para su programación, a su vez permite la simulación de los movimientos del robot desde el ordenador.
Utiliza el lenguaje Rapid, que se trata de un lenguaje de programación textual de alto nivel. Divide el programa en un programa principal y en módulos del sistema.

4.4.1 DIAGRAMA DE FLUJO



4.4.2 ARRANQUE ROBOT IRB 140

Para arrancar el robot IRB 140 en primer lugar se cambiará el botón de arranque de *OFF* a *ON*, que se encuentra en el armario del robot.

Ilustración 29. Botón ON

También se debe girar la llave hasta la posición de control manual de manera que los motores pasen a estado ON.

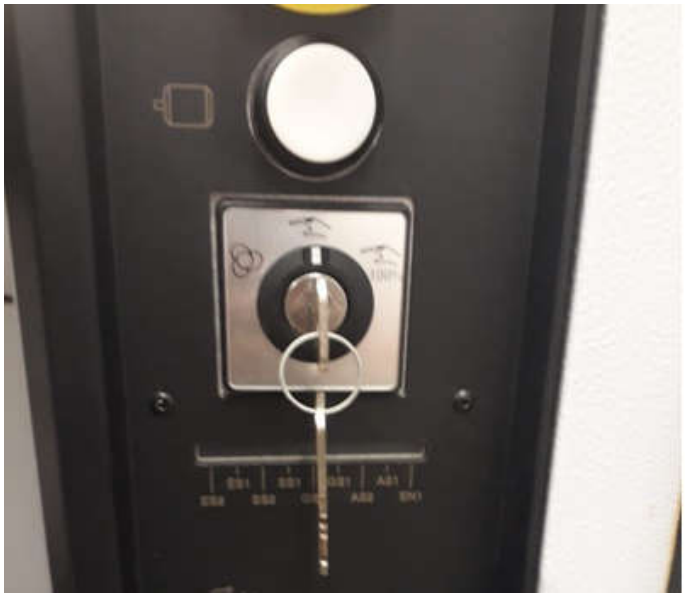


Ilustración 30. Llave opción manual

4.4.3 CONFIGURACIÓN ROBOT STUDIO

En primer lugar, se debe seleccionar la *solución con estación y controlador de robot*, el modelo de robot que en nuestro caso es el IRB 140 y pulsar el botón *crear*.

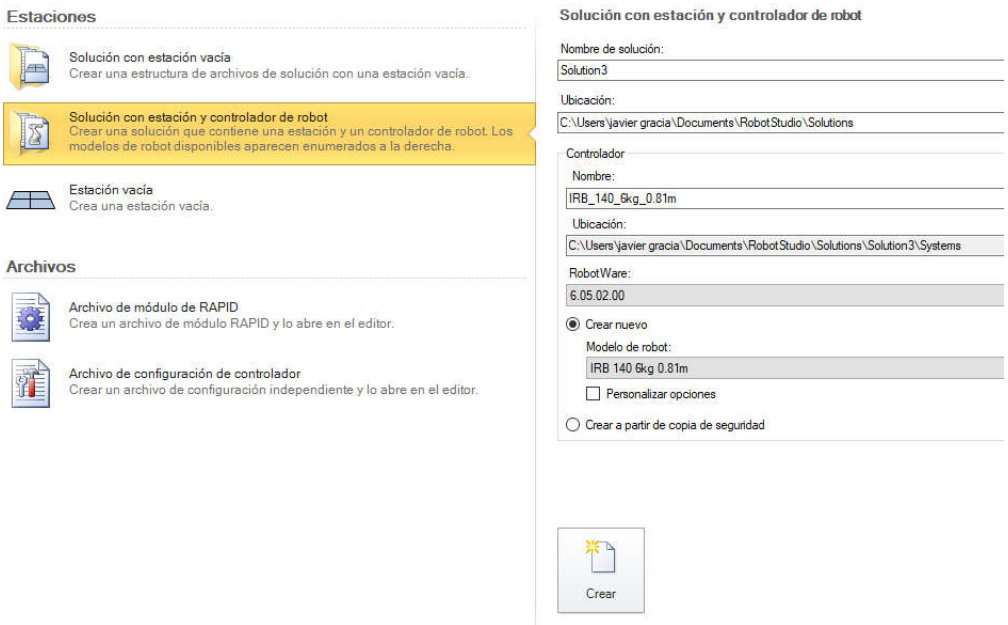


Ilustración 31. Configuración Robot Studio

A continuación, se selecciona la biblioteca para nuestro robot y pulsamos botón aceptar. No confundir con el modelo IRB 140T.

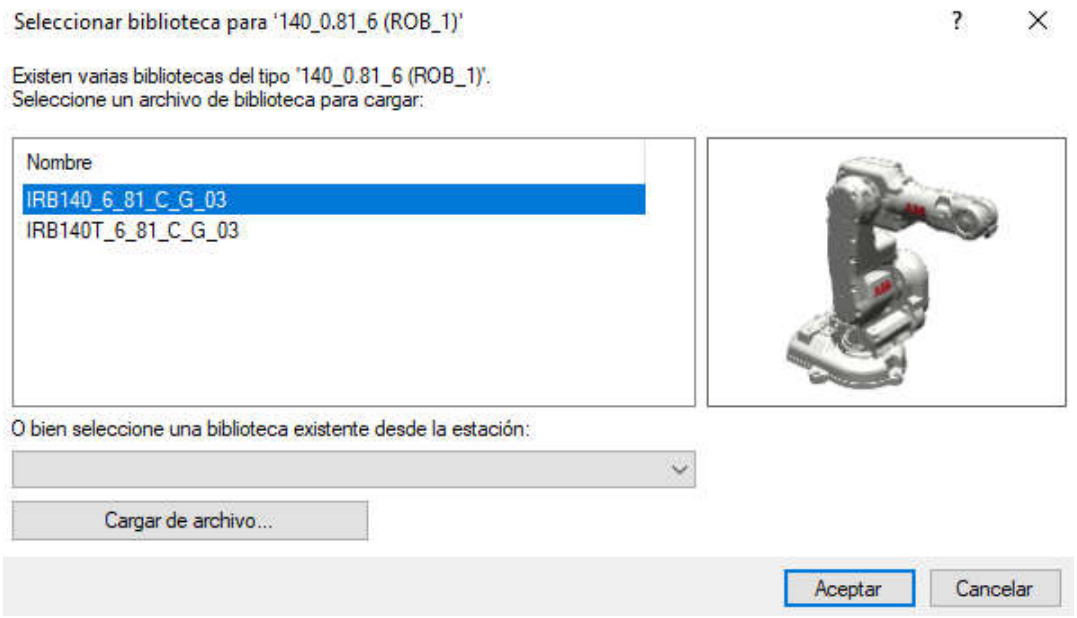


Ilustración 32. Biblioteca IRB 140

Una vez pulsado el botón de aceptar se selecciona en el menú la pestaña de Rapid y se pulsa el botón *main* para poder programar el robot con el lenguaje Rapid.

Se puede borrar los módulos de sistema *BASE* y *user*.

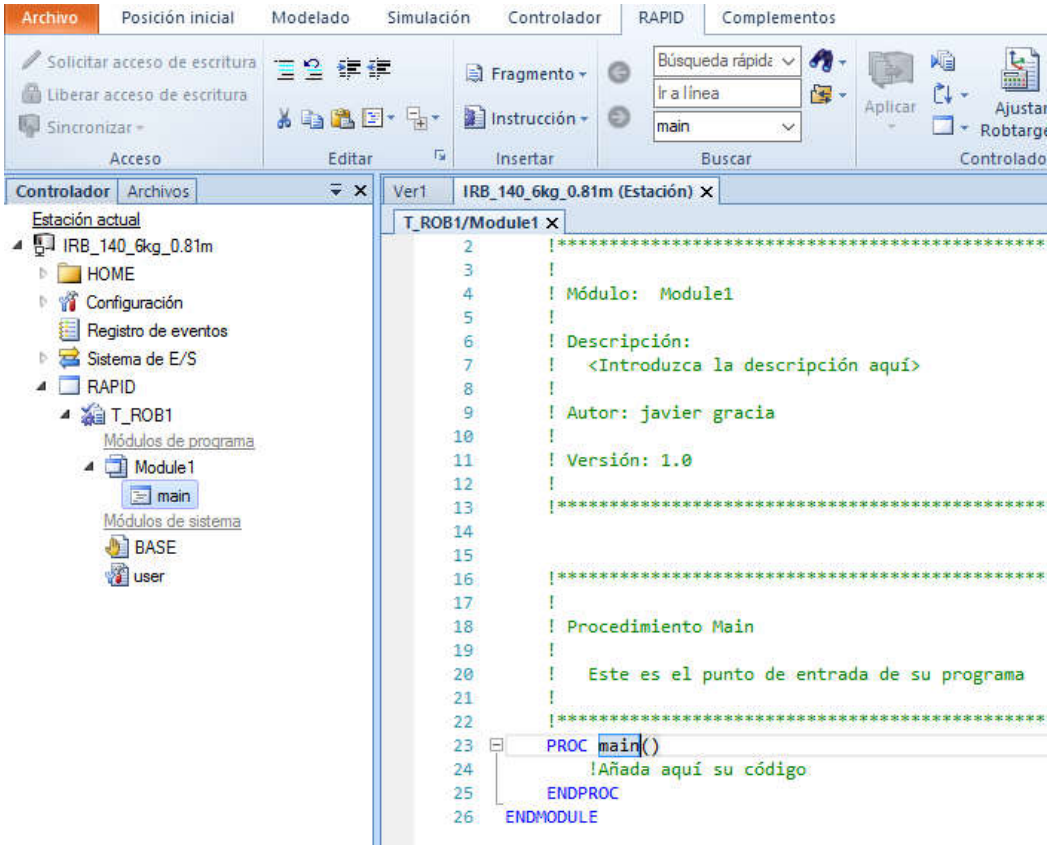


Ilustración 33. Modulo Main Rapid

4.4.4 PROGRAMACIÓN CON LENGUAJE RAPID

La programación de Rapid consta de declaración de las variables, declaración de los *robtarget* y del programa principal, que se encarga de la comunicación con Sherlock 7 y de programar los movimientos para coger la pieza y dejarla.
Para dejar la pieza se distinguirá, como hemos comentado anteriormente, por la posición de la lata sobre la cinta.
En el caso de que la obertura este visible, el robot llevará la pieza al inicio del proceso industrial.
Por el contrario si la pieza tiene la obertura oculta, el robot llevará la pieza a un sistema el cual consiga girar la lata para dejar la obertura visible.

El proceso industrial y el sistema para girar la lata no se implementarán en el trabajo.

4.4.4.1 DECLARACIÓN DE LAS VARIABLES

Las variables se declaran al inicio del programa de Rapid y se inicializarán para evitar errores.

4.4.4.2 DECLARACIÓN ROBTARGET

Los *robtarget* configuran la posición a la que llegará el robot y la de sus ejes externos.

Tiene la estructura siguiente:

" CONST robtarget punto :=
[[x,y,z],[q1,q2,q3,q4],[confEje1,confEje2,confEje3,ConfEjex],[9E9,

9E9,9E9,9E9,9E9,9E9]];".

En primer lugar, se declara las coordenadas donde queremos que llegue el robot.

En segundo lugar, se declaran los *quaternios*, la orientación del vector del punto.

En tercer lugar, se configuran los ejes.

Por último, se especifica el número de ejes externos que tengamos, en nuestro caso seis.

Para nuestro programa utilizaremos cuatro *robtarget*, todos ellos de tipo constante *CONST*. Uno de ellos se utiliza para llevar el robot a la posición de origen, otro para situar el robot encima del sensor de la cinta y los dos últimos para llevar el robot a la posición donde se dejará tanto la lata con la obertura visible como oculta.

4.4.4.3 COMUNICACIÓN CON SHERLOCK 7

La comunicación con Sherlock 7 se realizará mediante *Socket*, los cuales nos permiten transmitir información entre los programas y permiten la programación por separado de ambos.

En primer lugar se debe abrir el canal por el cual vamos a comunicarnos y se declara en las variables.

Para declarar el canal de comunicación se utilizará la instrucción *socketdev* y se declara como una variable. La instrucción quedará de la siguiente manera:
"VAR socketdev canal1;".

Para crear el canal se utiliza la instrucción: "SocketCreate canal1;".

Para la conexión con el programa Sherlock 7 se utilizará la instrucción: "SocketConnect canal1, "158.42.16.207", 1027;". El número señalado en rojo se trata de la dirección IP del Sherlock 7 y el número que le sigue tras la coma se trata del puerto.

Una vez tengamos todas la instrucciones anteriores escritas se puede iniciar el dialogo con Sherlock 7, para ello se utilizarán dos instrucciones.

La primera de ellas nos permite mandar a Sherlock 7 cadenas de caracteres, se trara de la instrucción: "SocketSend canal1\Str:="Hola!";", la cadena de caracteres enviada se trata de la parte que queda entre comillas de color rojo.

Para recibir las cadenas de caracteres enviadas por Sherlock 7 se utiliza la instrucción: "SocketReceive canal1\Str:=cadena_de_datos_recibida;", la cadena de caracteres recibida se guardará en la variable *cadena_de_datos_recibida*.

4.4.4.4 MOVIMIENTOS DEL ROBOT

Para realizar los movimientos del robot tanto para agarrar como para dejar la lata se usaran varias instrucciones.

Para realizar movimientos rápidos, se utiliza la instrucción MOVEJ, realiza el movimiento no lineal y los ejes llegan a la posición de destino a la vez. La instrucción tiene las siguientes características:
"MoveJ pos_origen,v100,fine,tool0;" .
En primer lugar, se escribe la posición a la que queremos llegar.
En segundo lugar, se determina la velocidad.
En tercer lugar, se especifica la precisión.
En último lugar, se escribe el nombre de la herramienta.

Los movimientos para coger la lata y dejarla a partir de las posiciones especificadas en los *robtarget* se realizan con la instrucción MOVEL. Esta

instrucción realiza el movimiento lineal hacia el destino. Tiene las siguientes características:

"MoveL offs(pCinta,posx,posy,-243.3),v20,fine,tool0;" .

En este caso utilizaremos el comando offs que nos permite escribir la posición X, Y y Z desde una posición determinada, en este caso *pCinta*.

4.4.4.5 MOVIMIENTOS DE LA PINZA

La pinza se encarga de coger la lata y dejarla mediante tres dedos que se abren y cierran. Para abrirlos y cerrarlos se utilizan distintas instrucciones.

La primera de ellas se utiliza para habilitar la pinza, la instrucción es así:
" Set GRIPPER_ENABLE;".

Para abrir la pinza se utiliza la instrucción: "Set GRIPPER_OPEN;".
Para cerrarla se utiliza: "Reset GRIPPER_OPEN;".

4.4.4.6 CÓDIGO RAPID

```
MODULE MainModule

 !Declaración de variables

VAR socketdev canal1;
 VAR string cadena_de_datos_recibida:="";
 VAR num longitud_trama:=0;
 VAR num posición_string_inicial:=0;
 VAR num posicion_string_final:=0;
 VAR num vuelta;
 VAR string v:="";
 VAR num posx:=0;
 VAR num posy:=0;
 VAR string trozo:="";
 VAR bool ok;
 VAR num ChPos1:=0;
 VAR num ChPos2:=0;
 VAR num Len2:=0;
 VAR num Cinta:=0;
```

```
!Declaración de los robtarget

CONST robtarget
pos_origen:=[[444.5,-
27,662.5],[0,0,1,0],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget
pCinta:=[[0,0,560],[0.04653,-0.01564,-0.9965,-
0.06747],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget
pLatas:=[[86.7,423.4,612],[0,0,1,0],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+0
9]];

CONST robtarget
pLatasMal:=[[418.3,358.6,612],[0.05686,0.04552,-
0.99519,0.06548],[0,2,2,1],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];


!Programa principal

PROC main()

!Mover robot a la posición de origen
MoveJ pos_origen,v100,fine,tool0;

!Borro Flexpendant
TPErase;

!Comunicación con Sherlock 7

SocketCreate canal1;
SocketConnect canal1, "158.42.16.207", 1027;
SocketSend canal1\Str:="Hola!";
SocketReceive canal1\Str:=cadena_de_datos_recibida;
SocketSend canal1\Str:="Estoy_listo!";
WaitTime 0.25;
TPWrite "                    ";
TPWrite "Conexion_Ok";
TPWrite cadena_de_datos_recibida;
TPWrite "                    ";
```

```
!Programa principal
WHILE TRUE DO

Set CONVEYOR_FWD;

WaitDI CONVEYOR_OBJ_SEN,1;

Reset CONVEYOR_FWD;

SocketSend canal1\Str:="cinta parada!";

de string, reconocimiento de posiciones y movimientos del robot

!Envía un mensaje de confirmación a sherlock para que analice la imagen

SocketSend canal1\Str:="Enviame datos!";
SocketReceive canal1\Str:=cadena_de_datos_recibida;
TPWrite cadena_de_datos_recibida;
longitud_trama:=StrLen(cadena_de_datos_recibida);
TPWrite "Longitud trama="\Num:=longitud_trama;



!Reconocimiento de la string
IF longitud_trama>0 THEN

posición_string_inicial:=0;
posicion_string_final:=1;
vuelta:=0;

WHILE vuelta<2 DO
ChPos1:=posición_string_inicial+posicion_string_final;
v:=StrPart(cadena_de_datos_recibida,ChPos1,1);

!Mientras sea distinto de "," leemos varbiale

WHILE v<>"," AND v<>"@" AND v<>"#" DO
posicion_string_final:=posicion_string_final+1;
ChPos1:=ChPos1+1;
v:=StrPart(cadena_de_datos_recibida,ChPos1,1);
ENDWHILE

Len2:=posicion_string_final-1;
ChPos2:=posición_string_inicial+1;
```

```
 trozo:=StrPart(cadena_de_datos_recibida,ChPos2,Len2);

IF (vuelta MOD 2)=0 THEN
ok := StrToVal(trozo,posx);
ELSE
ok := StrToVal(trozo,posy);
ENDIF

 !guarda variable x y escribe variable y

 IF v="," THEN
 posición_string_inicial:=posición_string_inicial+posicion_string_final;
 posicion_string_final:=1;


 ENDIF
 vuelta:=vuelta+1;
 ENDWHILE
 TPWrite "Posicion x="\Num:=posx;
 TPWrite "Posicion y="\Num:=posy;
Set GRIPPER_ENABLE;
 !Movimiento para coger lata correcta
  MoveL offs(pCinta,posx,posy,0),v100,fine,tool0;
Set GRIPPER_OPEN;
 !ir a posicion cinta
MoveL offs(pCinta,posx,posy,-243.3),v20,fine,tool0;
Reset GRIPPER_OPEN;
 !ir a posicion intermedia cinta
MoveL offs(pCinta,posx,posy,0),v50,fine,tool0;



IF v="@" THEN
MoveJ pLatas,v200,fine,tool0;
MoveL offs(pLatas,0,0,-437.5),v50,fine,tool0;
Set GRIPPER_OPEN;
MoveJ pLatas,v200,fine,tool0;
ENDIF
IF v="#" THEN
MoveJ pLatasMal,v200,fine,tool0;
MoveL offs(pLatasMal,0,0,-438.5),v50,fine,tool0;
Set GRIPPER_OPEN;
MoveJ pLatasMal,v100,fine,tool0;
```

```
ENDIF

 !Mover robot a la posición de origen
 MoveJ pos_origen,v200,fine,tool0;

ENDIF
 ENDWHILE
 SocketClose canal1;
ENDPROC


      ENDMODULE
```

4.5 CARGA DE PROGRAMA RAPID EN ROBOT IRB 140

Para cargar el programa de Rapid en el robot IRB 140 previamente se debe cargar la carpeta que contenga el programa en una unidad USB.

Lo siguiente que se debe hacer es introducir la unidad USB en el puerto USB que se encuentra en el armario robot.

Ilustración 34. Puerto Usb Armario robot

Desde la consola *FlexPendant* se ha de acceder a la ventana de producción que se encuentra en el menú ABB.

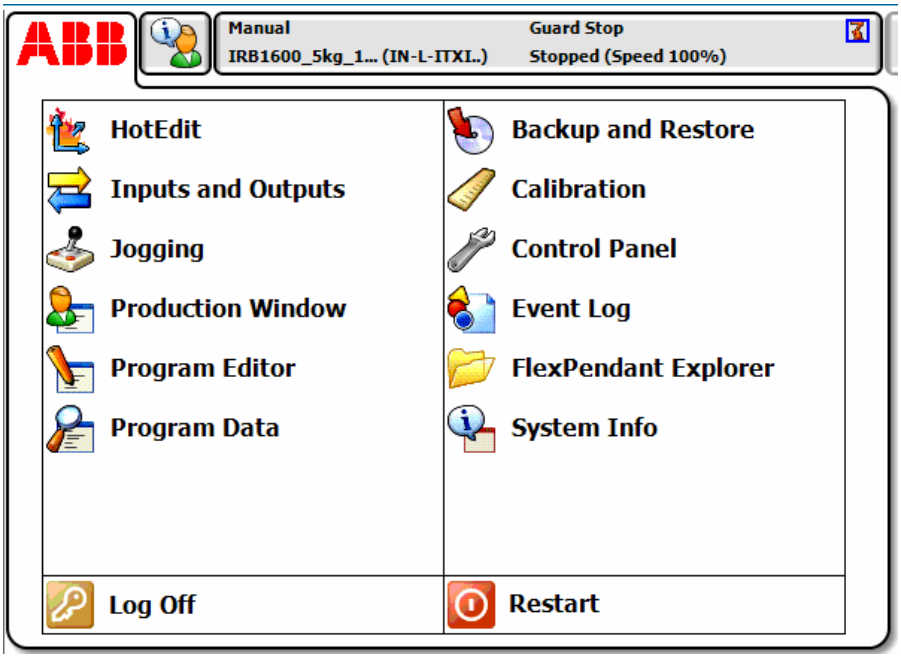


Ilustración 35. *Menu ABB* Flex Pendant [1]

A continuación se debe que cargar el programa para su ejecución pulsando en el botón *cargar programa*.

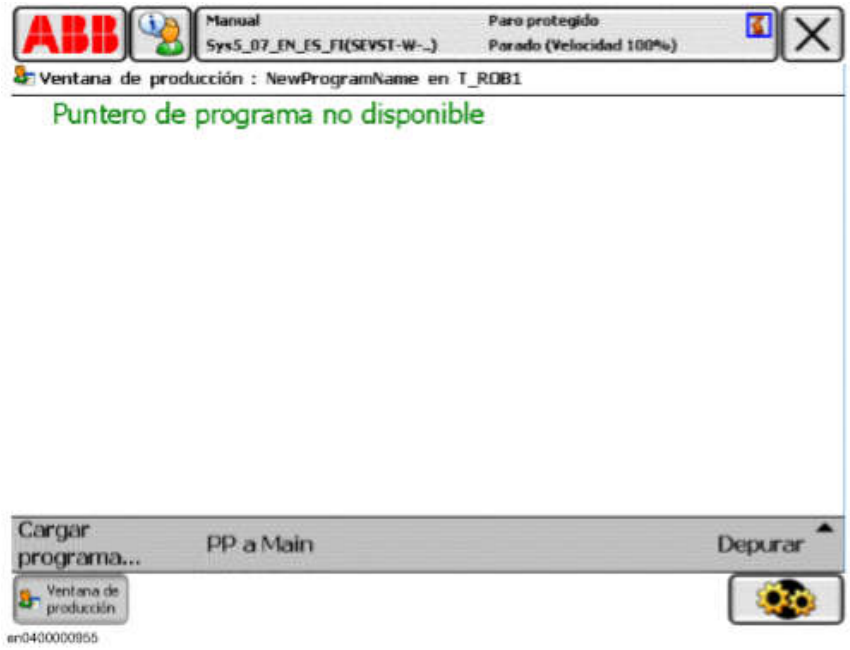


Ilustración 36. *Ventana de producción* Flex Pendant [2]

Por último, una vez cargado el programa, basta con pulsar el botón *play* que se encuentra en la propia consola y mientras se mantiene pulsado el botón de *dispositivo de habilitación.*

Ilustración 37. Botón *dispositivo de habilitación* Flex Pendant

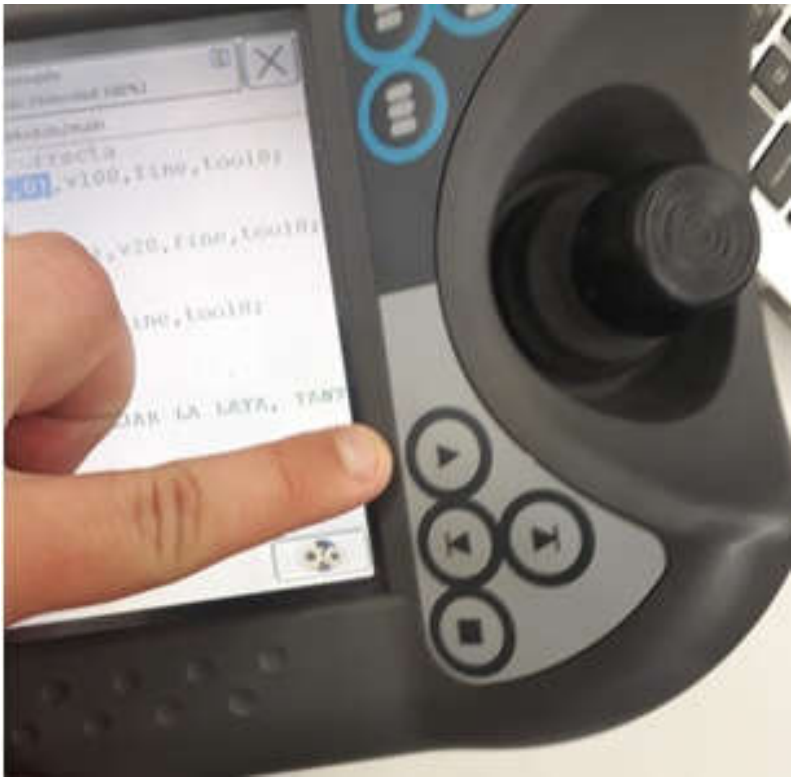


Ilustración 38. Botón Play Flex Pendant

5. NORMATIVA

La normativa vigente para los brazos robóticos es la siguiente:

- UNE EN ISO 10218-2:2011 Robots y dispositivos electrónicos. Requisitos de seguridad para robots industriales. Parte 2: Sistemas de robot e integración. [3]

-ISO/TS 15066 Robots y dispositivos electrónicos. Robots colaborativos. [4]

-ISO/TS 15066:2015 Robots y dispositivos robóticos. Robots de colaboración.[5]

- ISO/TS 15066:2016 Robots colaborativos. Marca las normas para el trabajo con robots colaborativos. [6]

-UNE-EN 775 ISO 10218 :1992 Robots manipuladores industriales. Seguridad. [7]

- UNE EN ISO 8373: 1998 Robots Manipuladores Industriales “Vocabulario”. Define al robot manipulador.

- UNE EN ISO 10218-1:2011 Robots y dispositivos electrónicos. Requisitos de seguridad para robots industriales. Parte 1: Robots.

6.CONCLUSIONES

El uso de la visión artificial proporciona una innumerable cantidad de soluciones para afrontar los problemas que nos presentan los proyectos.

Para asegurar el éxito de las soluciones se ha de tener una buena iluminación de la zona de trabajo ya que es muy sensible a los brillos y sombras que dificultan las soluciones y da errores. En el caso del laboratorio donde se ha realizado el proyecto la luz natural del exterior afectaba a la solución aplicada.
Este problema se ha solucionado ajustando los parámetros del programa Sherlock 7 de forma que funcione de forma correcta independientemente de los cambios de luz que ocasiona el paso de las horas del día.

Una vez solucionados los problemas que presentaba el proyecto se puede concluir que:

-Se ha conseguido el reconocimiento de la pieza utilizando el programa Sherlock 7.

-Se ha podido comunicar el programa Sherlock 7 con el robot ABB modelo IRB 140 utilizando Sockets, que a su vez permiten realizar los dos programas por separado.

-Se ha podido separar la información recibida por el programa Sherlock 7 con el código de Rapid de forma que se ha podido guardar en variables la coordenada X, la coordenada Y y la posición de la lata sobre la cinta.

-Se han programado los movimientos tanto para coger y dejar la pieza como para agarrar la pieza y soltarla de forma satisfactoria.

7.BIBLIOGRAFÍA

Manual Flex Pendant: http://isa.uniovi.es/~jalvarez/abb/es/3HAC16590-es.pdf [1] [2]

Manual IRB 140: http://isa.uniovi.es/~jalvarez/abb/es/3HAC027400-es.pdf

Nomrativa ISO: http://www.infoplc.net/actualidad-industrial/item/103207-iso-norma-ts15066-robots-colaborativos [3] [4] [5]

Normativa ISO: http://www.infoplc.net/actualidad-industrial/item/103207-iso-norma-ts15066-robots-colaborativos [6]

Normativa ISO: https://www.researchgate.net/profile/Giuseppe_Carbone/publication/237659045_CRITERIOS_PARA_LA_SEGURIDAD_EN_EL_USO_DE_ROBOTS/links/00463527a05fef124b000000/CRITERIOS-PARA-LA-SEGURIDAD-EN-EL-USO-DE-ROBOTS.pdf [7]

ETSID

UNIVERSIDAD POLITÉCNICA DE VALENCIA

PROGRAMACIÓN DE UNA CÉLULA ROBOTIZADA PARA LA MANIPULACIÓN DE LATAS DE CONSERVA

DOCUMENTO:

PRESUPUESTO

Autor: Javier Gracia Andrés

Tutor: Carlos Ricolfe Viala

Titulación: Grado en Ingeniería Electrónica Industrial y Automática

ÍNDICE

ÍNDICE DE TABLAS

PRESUPUESTO

En este apartado se pretende detallar el presupuesto de los sistemas hardware, sistemas software y mano de obra para desarrollar el proyecto.

1. SISTEMAS HARDWARE

Los elementos hardware son los elementos físicos como el robot IRB 140, la cámara de visión artificial, etc.

ELEMENTO	COSTE/UNIDAD (€/UDS.)	CANTIDAD (UDS.)	COSTE TOTAL (€)
ROBOT IRB 140	19000,00	1	19000,00
CÁMARA JAI CV-M77	350,00	1	350,00
ORDENADOR INDUSTRIAL INFAMOIN	700,00	1	700,00
CINTA TRANSPORTADORA	450,00	1	450,00
CPU	532,00	1	532,00
MONITOR	129,50	1	129,50
RATON	10,25	1	10,25
TECLADO	14,99	1	14,99
TOTAL			20486,74

Tabla 1. Sistemas Hardware

2. SISTEMAS SOFTWARE

Las licencias de Sherlock 7 y Robot Studio 6.05.02 tienen un tiempo de uso anual.

ELEMENTO	COSTE/UNIDAD (€/UDS.)	CANTIDAD (UDS.)	COSTE TOTAL (€)
WINDOWS 10	39,30	1	39,30
SHERLOCK 7	65,00	1	65,00
ROBOT STUDIO 6.05.02	72,00	1	72,00
TOTAL			176.30

Tabla 2. Sistemas Software

3. MANO DE OBRA

En este apartado se calculará el coste que supone el tiempo dedicado por parte del ingeniero en realizar el proyecto.

El ingeniero percibe un sueldo base de 1300 €/mes, teniendo en cuenta que dedica 8 horas diarias el coste por hora de 8,125 €/h.

Para realizar este presupuesto no se ha tenido en cuenta el tiempo dedicado a la instalación de los equipos.

OPERACIONES	SALARIO (€/h.)	CANTIDAD (h.)	COSTE TOTAL (€)
ESTUDIO	9,10	20	182,00
DISEÑO	9,10	70	637,00
IMPLEMENTACIÓN	9,10	45	409,05
COMPROBACIÓN	9,10	30	273,00
TOTAL			1501,05

Tabla 3. Mano de obra

4. TOTAL PRESUPUESTO

El total será la suma de los tres presupuestos anteriores.

20486,74 €+176,30 €+1501,05 €=22164,09 €

Siendo el total 22164,09 €.